Attorney's Docket No.: 10559-612001 / P12851

AF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Gilbert Wolrich et al.   Art Unit : 2166
Serial No. : 10/039,289   Examiner : Srirama Channavajjala
Filed   : January 4, 2002
Title   : QUEUE ARRAYS IN NETWORK DEVICES

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

<u>APPEAL BRIEF ON BEHALF OF GILBERT WOLRICH ET AL.</u>

Enclosed is a $500 check for the Appeal Brief fee and a $120 check for the Petition for Extension of Time fee. Please apply any other charges or credits to Deposit Account No. 06-1050, referencing Attorney's Docket No. 10559-612001.

(1)   Real Party in Interest

The real party in interest in the above application is Intel Corporation.

(2)   Related Appeals and Interferences

The appellant Appellant is not aware of any appeals or interferences related to the above-identified patent application.

(3)   Status of Claims

This is an appeal from the decision of the Primary Examiner in an Office Action dated September 28, 2005, rejecting claims 1-37, all of the claims of the above application. The claims have been twice rejected. Claims 1-37 are the subject of this appeal.

(4)   Status of Amendments

All amendments have been entered. Appellant filed a Notice of Appeal on **February 14, 2005**. Appellant filed a Pre-Appeal Brief Request for Review concurrently with the Notice of Appeal. The panel determined, in an Office communication mailed March 28, 2006, that the application should proceed to the Board of Patent Appeals and Interferences because there is at least one actual issue for appeal.

(5)   Summary of Claimed Subject Matter

Background

The claimed invention relates to queue arrays in network devices. [Specification page 1, lines 4-5]

Appellant's Invention

Claim 1

One aspect of Appellant's invention is set out in claim 1 as a method.

Figure 2

Applicant : Gilbert Wolrich et al.
Serial No. : 10/039,289
Filed : January 4, 2002
Page : 3 of 24

Attorney's Docket No.: 10559-612001 / P12851

Inventive features of Claim 1 include: "storing in memory a plurality of queue descriptors each including a head pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last element in the corresponding queue." Appellant's FIG. 3 (reproduced below) shows an example of an output queue 18 and its corresponding queue descriptor. The output queue 18 includes a linked list of elements 22, each of which contains a pointer 24 to the next element 22 in the output queue 18. The queue descriptor 20 includes a head pointer 28, a tail pointer 30 and a count 32. The head pointer 28 points to the first element 22 of the output queue 18, and the tail pointer 30 points to the last element 22 of the output queue 18. The count 32 identifies the number (N) of elements 22 in the output queue 18. [Specification page 3, line 6 – line 25]
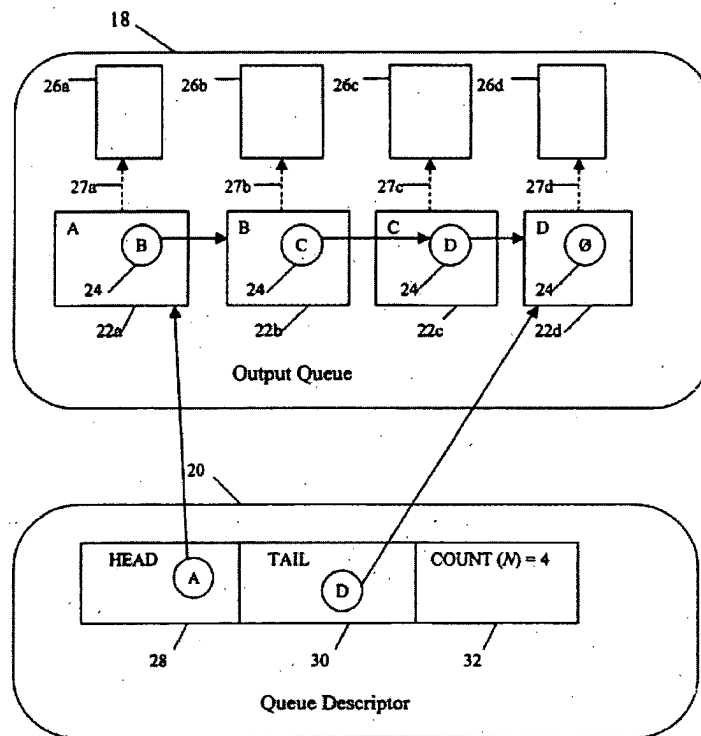

Figure 3

Inventive features of Claim 1 also include: "in response to a command to perform an enqueue or dequeue operation with respect to a first queue, fetching from the memory to a cache one of either the head pointer or tail pointer of a first queue descriptor corresponding to the first queue." In order to reduce the read and write operations between the cache 12 and the memory

14, it is possible to fetch and return only those parts of the queue descriptor 20 necessary for the enqueue or dequeue operations. [Specification page 4, line 17 – line 20] For example, the head pointer 28 need not be fetched during an enqueue operation, thereby saving read bandwidth between the processor 10 and memory 14. Similarly, a tail pointer 30 need not be fetched from memory 14 during a dequeue operation. [Specification page 8, line 3 – line 7]

Inventive features of Claim 1 also include: "returning to the memory from the cache portions of the first queue descriptor modified by the operation." When a queue descriptor 20 is removed 42, 81 from the cache 12, the processor 10 checks the valid bits 34, 36. If there were no modifications to the tail pointer 30 (for example, when only dequeue operations were performed on the queue), the tail pointer valid bit 36 remains unset. This indicates that write bandwidth can be saved by writing back to memory 14 only the count 32 and head pointer 28. If there were no modifications to the head pointer 28 (for example, when only enqueue operations to a non-empty output queue 18 were performed), the head pointer valid bit 34 remains unset. This indicates that only the count 32 and tail pointer 30 need to be written back to the queue descriptor 20 in memory 14, thus saving write bandwidth. [Specification page 8, line 7 – line 19]

Claim 7

Another aspect of the invention is a method as recited in claim 7.

Inventive features of Claim 7 include: "storing in memory a plurality of queue descriptors each including a head pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last element in the corresponding queue [Specification page 3, line 20 – line 24]; determining whether a head pointer or a tail pointer of a queue descriptor that was fetched from the memory to a cache in response to an enqueue or dequeue operation had been modified by the enqueue or dequeue operation; and returning one of either the head pointer or tail pointer to the memory from the cache only if that pointer had been modified. [Specification page 8, line 9 – line 19]"

Claim 14

Applicant : Gilbert Wolrich et al.
Serial No. : 10/039,289
Filed : January 4, 2002
Page : 5 of 24

Attorney's Docket No.: 10559-612001 / P12851

Another aspect of the invention is covered by Claim 14 directed to an apparatus [Specification page 2, line 6 – page 3, line 5].

Inventive features of Claim 14 include "memory for storing a plurality of queue descriptors, each of which includes a head pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last element in the corresponding queue [Specification page 3, line 20 – line 24]; a cache for storing queue descriptors corresponding to up to a number of the memory's queue descriptors [Specification page 4, line 4]; and a processor configured to: fetch from the memory to the cache one of either the head pointer or the tail pointer of a particular queue descriptor in response to a command to perform an enqueue or a dequeue operation with respect to the particular queue descriptor [Specification page 4, line 17 – line 20]; and return to the memory from the cache portions of the queue descriptor modified by the operation. [Specification page 8, line 9 – line 19]"

Claim 26

Another aspect of the invention is covered by Claim 26 directed to an article comprising a computer-readable medium that stores computer-executable instructions [Specification page 9, line 14 – line 25].

Inventive features of claim 26 include instructions to "store in memory a plurality of queue descriptors each including a head pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last element in the corresponding queue [Specification page 3, line 20 – line 24]; in response to a command to perform an enqueue or dequeue operation with respect to a first queue corresponding to a first queue descriptor, fetch from memory to a cache one of either a head pointer of the first queue descriptor pointing to a first element in the first queue or a tail pointer of the first queue descriptor pointing to a last element in the first queue [Specification page 4, line 17 – line 20]; and return to the memory from the cache the portions of the first queue descriptor modified by the operation. [Specification page 8, line 9 – line 19]"

Claim 31

Another aspect of the invention is covered by Claim 31 directed to a method.

Inventive features of Claim 31 include: "storing in memory a queue descriptor for a queue, the queue descriptor including a count identifying a number of elements in the queue, a head pointer pointing to a first element in the queue and a tail pointer pointing to a last element in the queue [Specification page 3, line 20 – line 25]; in response to a command to perform an enqueue or dequeue operation with respect to the queue, fetching from the memory to a cache the count and one of either the head pointer or tail pointer [Specification page 4, line 17 – line 20]; and returning to the memory from the cache portions of the queue descriptor modified by the operation. [Specification page 8, line 9 – line 19]"

Claim 33

Another aspect of the invention is covered by Claim 33 directed to a method. Inventive features of claim 33 include: "determining whether a head pointer or a tail pointer of a queue descriptor that was fetched from memory to a cache in response to an enqueue or dequeue operation on a queue had been modified by the enqueue or dequeue operation; and returning a count identifying a number of elements in the queue [Specification page 3, line 20 – line 25] and one of either the head pointer or tail pointer to the memory from the cache only if that pointer had been modified. [Specification page 8, line 9 – line 19]"

Claim 34

Another aspect of the invention is covered by claim 34. Claim 34 is directed to an apparatus [Specification page 2, line 6 – page 3, line 5]. Inventive features of Claim 34 include: "memory for storing queue descriptors which include a count identifying a number of elements in a queue, a head pointer pointing to a first element in the queue and a tail pointer pointing to a last element in the queue [Specification page 3, line 20 – line 25]; a cache for storing queue descriptors corresponding to up to a number of the memory's queue descriptors [Specification page 4, line 4]; and a processor configured to: fetch from the memory to the cache the count and one of either the head pointer or the tail pointer of a particular queue descriptor in response to a command to perform an enqueue or a dequeue operation with respect to the particular queue descriptor [Specification page 4, line 17 – line 20]; and return to the memory from the cache

portions of the queue descriptor modified by the operation. [Specification page 8, line 9 – line 19]"

Claim 36

Another aspect of the invention is covered by claim 36. Claim 36 is directed to an article comprising a computer-readable medium that stores computer-executable instructions [Specification page 9, line 14 – line 25]. Inventive features of claim 36 include instructions to: "store in memory a queue descriptor including a count identifying a number of elements in a queue, a head pointer pointing to a first element in the queue and a tail pointer pointing to a last element in the queue [Specification page 3, line 20 – line 25]; in response to a command to perform an enqueue or dequeue operation with respect to a queue, fetch from memory to a cache the count and one of either a head pointer pointing to a first element in the queue or a tail pointer pointing to a last element in the queue [Specification page 4, line 17 – line 20]; and return to the memory from the cache the portions of the queue descriptor modified by the operation. [Specification page 8, line 9 – line 19]"

(6)   Ground of Rejection

Claims 1-37 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Pascal, EP 0760501, in view of Slane, U.S. Patent No. 6,438,651.

(7)   Argument

Obviousness

"It is well established that the burden is on the PTO to establish a *prima facie* showing of obviousness," *In re Fritsch*, 972 F.2d. 1260, 23 U.S.P.Q.2d 1780 (C.C.P.A., 1972).

"It is well established that there must be some logical reason apparent from the evidence or record to justify combination or modification of references." *In re Regal*, 526 F.2d 1399 188, U.S.P.Q.2d 136 (C.C.P.A. 1975). "In addition, even if all of the elements of claims are disclosed in various prior art references, the claimed invention taken as a whole cannot be said to be obvious without some reason given in the prior art why one of ordinary skill in the art would

Applicant : Gilbert Wolrich et al.
Serial No. : 10/039,289
Filed : January 4, 2002
Page : 8 of 24

Attorney's Docket No.: 10559-612001 / P12851

have been prompted to combine the teachings of the references to arrive at the claimed invention. Id. Even if the cited references show the various elements suggested by the Examiner in order to support a conclusion that it would have been obvious to combine the cited references, the references must either expressly or impliedly suggest the claimed combination or the Examiner must present a convincing line of reasoning as to why one skilled in the art would have found the claimed invention obvious in light of the teachings of the references." *Ex Parte Clapp*, 227 U.S.P.Q.2d 972, 973 (Board. Pat. App. & Inf. 985).

"The mere fact that the prior art could be so modified would not have made the modification obvious unless the prior art suggested the desirability of the modification." *In re Gordon*, 221 U.S.P.Q. 1125, 1127 (Fed. Cir. 1984).

> Although the Commissioner suggests that [the structure in the primary prior art reference] could readily be modified to form the [claimed] structure, "[t]he mere fact that the prior art could be so modified would not have made the modification obvious unless the prior art suggested the desirability of the modification." *In re Laskowski*, 10 U.S.P.Q. 2d 1397, 1398 (Fed. Cir. 1989).

"The claimed invention must be considered as a whole, and the question is whether there is something in the prior art as a whole to suggest the desirability, and thus the obviousness, of making the combination." *Lindemann Maschinenfabrik GMBH v. American Hoist & Derrick*, 221 U.S.P.Q. 481, 488 (Fed. Cir. 1984).

> Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Under Section 103, teachings of references can be combined only if there is some suggestion or incentive to do so. *ACS Hospital Systems, Inc. v. Montefiore Hospital*, 221 U.S.P.Q. 929, 933 (Fed. Cir. 1984) (emphasis in original, footnotes omitted).

"The critical inquiry is whether 'there is something in the prior art as a whole to suggest the desirability, and thus the obviousness, of making the combination.'" *Fromson v. Advance Offset Plate, Inc.*, 225 U.S.P.Q. 26, 31 (Fed. Cir. 1985).

Discussion

Claims 1, 7-14, 20-26, and 29-30

For the purposes of this appeal only, claims 1, 7-14, 20-26, and 29-30 may be treated as

standing or falling together. Claim 1 is representative of claims 1, 7-14, 20-26, and 29-30.

Claim 1 recites "storing in memory a plurality of queue descriptors each including a head

pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last

element in the corresponding queue; in response to a command ... with respect to a first queue,

fetching from the memory to a cache one of either the head pointer or tail pointer of a first queue

descriptor corresponding to the first queue."

Assuming *arguendo* that a person of ordinary skill in the art was motivated to combine

the teachings of Pascal and Slane, Appellant contends that no proper combination of these

references describes or suggests the claimed invention.

The Examiner cites col. 7, line 14-20, and FIG. 2-3 of Pascal to support the assertion that

Pascal teaches that each of a plurality of queue descriptors includes "a head pointer pointing to a

first element in a corresponding queue and a tail pointer pointing to a last element in the

corresponding queue." [Office Action dated September 28, 2005, page 4] However, this portion

of Pascal reproduced in context below (col. 7, lines 11-24) clearly does not support such an

assertion.

> Figure 3 shows a circular queue 10 formed in paged memory and used to
> buffer the transfer of data items between a producer entity 12 and a
> consumer entity 13. The overall arrangement shown in Figure 3 is similar
> to that of Figure 2 as regards the circular queue itself and the maintenance
> of head and tail pointers.
>
> Thus, in the Figure 3 arrangement, the producer entity 12 maintains a tail
> pointer in register 30, this pointer corresponding to the address in memory
> of the next storage location to be written to. This pointer is maintained
> within a queue segment (that is, for storage locations within the same
> memory page) by updating it each time a data item is written by the
> number of memory addresses occupied by each data item.

As is evident from the passage above and Figures 2 and 3, Pascal does not describe or

even suggest that there is more than one "queue descriptor" that includes both a head pointer and

a tail pointer. Instead, Pascal is concerned with maintaining a single set of head and tail pointers

for a single circular queue that has a plurality of segments. Claim 1 on the other hand requires

storing in memory a plurality of queue descriptors each including a head pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last element in the corresponding queue. Appellant contends that the Examiner's interpretation of Pascal amounts to nothing more than unstated modification of Pascal. This unstated modification however is completely unsupportable by any rational reading of the reference, since the reference is directed to a circular queue and the features of storing in memory a plurality of queue descriptors each including a head pointer ... and a tail pointer would be of no import to such a structure.

The Office Action cites col. 9, lines 1-15 of Pascal to support the Examiner's assertion that Pascal teaches "fetching from the memory to a cache one of either the head pointer or tail pointer of a first queue descriptor corresponding to the first queue." However, this passage, reproduced below, does not even disclose a cache, much less that a head or tail pointer is fetched from a memory to a cache.

> When the producer entity wants to pass a data item to the circular queue, it first checks the queue status. It does this by reading the C-Index value maintained by the consumer entity into the producer-entity register 36 (step50), and then comparing on a module N basis, the value (P-Index + 1) with the C-Index copy value in register 36 (step 51). If the compared values are equal the queue is full (as was explained above with reference to Figure 5) and the producer entity must wait.
>
> If, however, the compared values are not equal, the queue is not full and the producer entity 12 proceeds to write the data item to the next available data-item storage location as indicated by the tail pointer held in register 30 (step 52).

Instead, this passage describes "reading [a] C-Index value maintained by a consumer entity into a producer-entity register" to check the queue status. This description is neither equivalent to nor suggests "fetching from the memory to a cache one of either the head pointer or tail pointer of a ... queue descriptor corresponding to [a] queue," where the "head pointer [points] to a first element in a corresponding queue and a tail pointer [points] to a last element in the corresponding queue."

First, Pascal describes the C-Index and P-Index as distinct from the head pointer and tail pointer. For example, Pascal describes that "the consumer [entity] maintains in register 37 a C-

Index that represents the location in the queue next to be read from, in terms of a position in the notional continuum 40. The C-Index is incremented on a modulo N basis each time a data item is read from the queue 10" (col. 8, lines 13-18). Furthermore, with regard to the "notional continuum 40" Pascal states that "this continuum is illustrated at 40 in Figure 3 but it is stressed that this a notional element, not a real one" (col. 8, lines 2-3). Therefore, Appellant contends that the C-Index is not a pointer to a first (or last) element in the queue, but rather is a value associated with a notional element used in determining queue status (), which has no relevance to the claimed features of fetching one of either the head pointer or the tail pointer from the memory to a cache. As Pascal discloses at col. 8, lines 32-38:

> At this point it is appropriate to comment on the tests used for queue full and queue empty. In the present example, the queue 10 is said to be empty when the P-Index and C-Index represent the same position in the continuum 40 (this, in turn, corresponding to the head and tail pointers pointing to the same storage location in queue 10).

The P-Index and C-Index are values that merely represent a distance between the first and last elements in a queue.

Second, Pascal describes that: "The producer entity 12 is provided with a copy of the C-Index this copy being kept in register 36. The consumer entity 13 is provided with a copy of the P-Index, this copy being kept in register 39" (col. 8, lines 22-25). The teaching that the C-Index is sent from a register (37) of the consumer entity 13 to a register (36) of the producer entity 12 is neither equivalent to nor suggestive of a value being "fetched from the memory to a cache." To the contrary, one of ordinary skill in the art would not have been motivated to modify the entity-to-entity transfer described by Pascal to a memory-to-cache transfer considering the suggestion of Pascal that "the copying across of the P-Index to the consumer entity and the C-Index to the producer entity may be initiated by either the passing or receiving entity" (col. 11, lines 1-4).

Furthermore, Appellant contends that the Examiner has failed to provide a proper motivation to suggest the desirability of combining Pascal with Slane. Appellant contends that claim 1 is allowable over the cited art because the references themselves fail to suggest the desirability to combine their teachings, and the Examiner has not presented a cogent line of

reasoning as to why one of ordinary skill in the art, having the references before him, would modify and combine the teachings of Pascal and Slane to arrive at the claimed invention.

The Examiner's stated motivation for combining the teachings of Pascal and Slane is: "because that would have allowed users of Pascal to use enqueue dequeue operations with respect to queue not only optimizes access requests during subsequent access requests to the queue, but also read and write are assured as each subsequent read/dequeue and write/enqueue request as suggested by Slane [col 2, line 52-59] bringing the advantages of improving queue performance and accessing to the data." However, the cited passage reproduced in context below (Slane, col. 2, line 55 – col. 3, line 2) would not have motivated one of ordinary skill in the art to combine the teachings of Pascal and Slane.

> In general terms, the present invention facilitates queue status
> determination, by maintaining, in addition to write/read pointers pointing
> to physical address locations, an overall logical view of the queue in terms
> of indexes into a notional circular continuum of N storage locations (N
> being the total number of data-item storage locations in the circular queue).

While it is not exactly clear what the Examiner means by "read and write are assured as each subsequent read/dequeue and write/enqueue request," this description of "queue status determination" does not relate to what appears to be the Examiner's offered motivation of optimizing "access requests" and improving "queue performance." The description of "queue status determination" including maintaining an "overall logical view" of a queue would not have motivated one of ordinary skill in the art to combine Slane's teachings with those of Pascal, which describe a different queue structure and therefore different procedures for queue status determination.

Even if the teachings of Pascal and Slane were combined, the Examiner has not identified from either reference why one of ordinary skill in the art would be motivated to modify the combined teachings to arrive at the claimed invention. Appellant contends that the Examiner has not identified a teaching, suggestion, or motivation in either of the references as a basis to modify and combine the teachings to arrive at the claimed invention, or presented a cogent line of reasoning that would have suggested such modifications to those of ordinary skill in the art.

Applicant : Gilbert Wolrich et al.   Attorney's Docket No.: 10559-612001 / P12851
Serial No. : 10/039,289
Filed   : January 4, 2002
Page   : 13 of 24

Therefore, Appellant submits that the Examiner has not properly established a *prima facie* case of obviousness with respect to claim 1.

Claims 2-3, 15-16, and 27

For the purposes of this appeal only, claims 2-3, 15-16, and 27 may be treated as standing or falling together. Claim 27 is representative of claims 2-3, 15-16, and 27.

Claim 27 recites "instructions to cause the computer system to: fetch the head pointer and not the tail pointer … in response to a command to perform a dequeue operation … or fetch the tail pointer and not the head pointer … in response to a command to perform an enqueue operation," where the base claim requires the fetching to be "from memory to a cache."

The Examiner cites passages in Pascal (col. 7, lines 55-59; col. 8, line 1; col. 9, lines 11-15) and in Slane (col. 3, lines 52-55; col. 4, lines 47-51; col. 5, lines 61-63; fig. 3) with respect to claims 2-3, 15-16, and 27. While Pascal and Slane may describe reading a head pointer or tail pointer to determine a head or tail of a queue, neither Pascal nor Slane teaches or suggests fetching either the head pointer or tail pointer from any location to any other location in response to any command, much less, fetching as recited in claim 27. Slane describes writing data to a cache (col. 4, lines 45-47; col. 5, lines 61-63) and Slane also describes head and tail pointers (col. 3, lines 40-42, 52-55). However, in no sense does Slane suggest, much less describe writing the head or tail pointer to the cache. Appellant contends that the Examiner has made an unstated and improper modification of Slane to suggest that Slane supplies what is missing in Pascal, and uses this improper modification of Slane in order to modify Pascal to supply the missing feature of fetching the head (tail) pointer and not the tail (head) pointer from memory to a cache. Therefore, Appellant contends that no proper combination of these references describes or suggests the claimed invention recited in claim 27.

Claims 4-6, 17-19, and 28

For the purposes of this appeal only, claims 4-6, 17-19, and 28 may be treated as standing or falling together. Claim 28 is representative of claims 4-6, 17-19, and 28.

Claim 28 recites "instructions to cause the computer system to return to memory: the head pointer and not the tail pointer … if only dequeue operations are performed … the tail

pointer and not the head pointer … if only enqueue operations are performed … while the first queue is unempty; or both the head pointer and tail pointer of the first queue descriptor if both an enqueue and a dequeue are performed … or an enqueue operation was performed … while the first queue is empty."

The Examiner cites passages in Slane (col. 4, lines 51-52, 56-60; col. 5, lines 64-67; col. 6, lines 1-8; col. 7, lines 3-8; fig. 4-6) with respect to claims 4-6, 17-19, and 28. Not only do none of these passages teach or suggest returning to memory "the head pointer and not the tail pointer" or the "the tail pointer and not the head pointer" or "both the head pointer and tail pointer" according to the respective cases recited in claim 28, but the Examiner appears to have cited these passages based on words appearing in the passages without considering the actual meaning of the passages.

For example, the passage at col. 5, line 64 – col. 6, line 8 includes the words "return to" and "memory," but in passage reproduced below it is clear that nothing is being returned to memory, but rather a cache line includes data from main memory to be returned to subsequent sequential read requests:

> With the logic of FIGS. 4 and 5, a cache line used to service read requests includes the actual data from the main memory 60 to return to subsequent sequential read requests. However, for a line allocated to update operations, the data is not read into the cache line from main memory 60 because the data will eventually be overwritten during the sequential write/update operations. Thus, there is no need to actually prefetch and fill the line with data for lines allocated to the write operations. Furthermore, if there is overlap with the head line (i.e., when the tail approaches the head in the circular buffer), then the head line will already be loaded in cache 54.

As another example, the passage at col. 7, lines 3-8 includes the words "cache" and "both head and tail," but in the passage reproduced below it is clear that the passage is not saying that both the head pointer and tail pointer are returned to memory from the cache, but rather that both head and tail portions of the queue are stored in cache lines:

> In the preferred embodiments, at least two lines are allocated per queue, one for the enqueue/write operations and the other for dequeue/read operations. When the queue is small, a single line is adequate, but two lines are necessary as a queue grows because a cache line is needed for both the head and tail portions of the queue.

Therefore, Appellant submits that no proper combination of these references describes or suggests the claimed invention recited in claim 28.

Claims 31-37

For the purposes of this appeal only, claims 31-37 may be treated as standing or falling together. Claim 31 is representative of claims 31-37.

Claim 31 recites "storing in memory a queue descriptor for a queue, the queue descriptor including a count identifying a number of elements in the queue, a head pointer ... and a tail pointer ... in response to a command ... operation with respect to the queue, fetching from the memory to a cache the count and one of either the head pointer or tail pointer."

In rejection of claim 31 the Examiner fails to attempt to point out how the prior art references, alone or combined, teach or suggest "in response to a command to perform an enqueue or dequeue operation with respect to the queue, fetching from the memory to a cache the count and one of either the head pointer or tail pointer." Thus, the Examiner has omitted at least one of the three basic criteria needed for a *prima facie* §103(a) rejection of claim 31, namely "the prior art reference (or references when combined) must teach or suggest all the claim limitations."

The Examiner that Pascal teaches: "a count identifying a number of elements in the queue" [Office Action dated September 28, 2005, page 5] and cites column 6, lines 10-19 and column 8, lines 4-12. However, assuming *arguendo* that the Examiner's characterization of Pascal's teaching is correct, this teaching in Pascal does not suggest "fetching from the memory to a cache the count and one of either the head pointer or tail pointer," much less "[fetching] ... in response to a command to perform an enqueue or dequeue operation with respect to the queue." Neither does any proper combination of Pascal and Slane teach or suggest all the limitations of claim 31. Appellant submits that the Examiner has not properly established a *prima facie* case of obviousness with respect to claim 31.
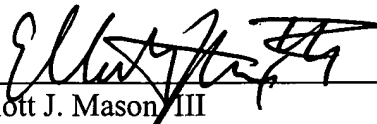
Conclusion

Appellant submits, therefore, that claims 1-37 are allowable over the cited art. Therefore, the Examiner erred in rejecting Appellant's claims and should be reversed.

Applicant : Gilbert Wolrich et al.
Serial No. : 10/039,289
Filed : January 4, 2002
Page : 16 of 24

Attorney's Docket No.: 10559-612001 / P12851

Respectfully submitted,

Date: _5-30-06_____

_____
Elliott J. Mason III
Reg. No. 56,569

Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

21305748.doc

Applicant : Gilbert Wolrich et al.
Serial No. : 10/039,289
Filed : January 4, 2002
Page : 17 of 24

Attorney's Docket No.: 10559-612001 / P12851

Appendix of Claims

1.   A method comprising:

storing in memory a plurality of queue descriptors each including a head pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last element in the corresponding queue;

in response to a command to perform an enqueue or dequeue operation with respect to a first queue, fetching from the memory to a cache one of either the head pointer or tail pointer of a first queue descriptor corresponding to the first queue; and

returning to the memory from the cache portions of the first queue descriptor modified by the operation.

2.   The method of claim 1 including fetching the head pointer and not the tail pointer of the first queue descriptor in response to a command to perform a dequeue operation with respect to the first queue.

3.   The method of claim 1 including fetching the tail pointer and not the head pointer of the first queue descriptor in response to a command to perform an enqueue operation with respect to the first queue.

4.   The method of claim 1 including returning to memory the head pointer and not the tail pointer of the first queue descriptor if only dequeue operations were performed on the first queue.

5.   The method of claim 1 including returning to memory the tail pointer and not the head pointer of the first queue descriptor if only enqueue operations were performed on the first queue while the first queue was unempty.

6.   The method of claim 1 including returning to memory the head pointer and tail pointer of the first queue descriptor if an enqueue and a dequeue operation were performed on

the first queue, or an enqueue operation was performed on the first queue while the first queue was empty.

7.   A method comprising:

storing in memory a plurality of queue descriptors each including a head pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last element in the corresponding queue;

determining whether a head pointer or a tail pointer of a queue descriptor that was fetched from the memory to a cache in response to an enqueue or dequeue operation had been modified by the enqueue or dequeue operation; and

returning one of either the head pointer or tail pointer to the memory from the cache only if that pointer had been modified.

8.   The method of claim 7 including using valid bits in the cache to track modifications to the pointers.

9.   The method of claim 8 including using a first valid bit to track modifications to the head pointer and second valid bit to track modifications to the tail pointer.

10.   The method of claim 9 including setting the first valid bit if a dequeue operation is performed with respect to the queue descriptor, or an enqueue operation is performed with respect to the queue descriptor while the queue is empty.

11.   The method of claim 9 including setting the second valid bit if an enqueue operation is performed with respect to the queue descriptor.

12.   The method of claim 9 including setting a pointer's valid bit when the pointer is fetched from the memory to the cache.

13.   The method of claim 9 including returning to the memory pointers whose valid bits have been set.

14.   An apparatus comprising:

memory for storing a plurality of queue descriptors, each of which includes a head pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last element in the corresponding queue;

a cache for storing queue descriptors corresponding to up to a number of the memory's queue descriptors; and

a processor configured to:

fetch from the memory to the cache one of either the head pointer or the tail pointer of a particular queue descriptor in response to a command to perform an enqueue or a dequeue operation with respect to the particular queue descriptor; and

return to the memory from the cache portions of the queue descriptor modified by the operation.

15.   The apparatus of claim 14 wherein the processor is configured to fetch the head pointer and not the tail pointer in response to a command to perform a dequeue operation.

16.   The apparatus of claim 14 wherein the processor is configured to fetch the tail pointer and not the head pointer in response to a command to perform an enqueue operation.

17.   The apparatus of claim 14 wherein the processor is configured to return to the memory the head pointer and not the tail pointer when only dequeue operations were performed on the queue.

18.   The apparatus of claim 14 wherein the processor is configured to return to the memory the tail pointer and not the head pointer if only enqueue operations were performed on the queue while the queue was unempty.

19. The apparatus of claim 14 wherein the processor is configured to return to the memory the head pointer and tail pointer if an enqueue and a dequeue operation were performed on the queue, or an enqueue operation was performed on the queue while the queue was empty.

20. The apparatus of claim 14 wherein the cache stores valid bits and wherein the processor is configured to track modifications to the pointers in the cache by setting the valid bits.

21. The apparatus of claim 20 wherein the cache stores a first valid bit to track modifications to the head pointer and a second valid bit to track modifications to the tail pointer.

22. The apparatus of claim 21 wherein the processor is configured to set the first valid bit if a dequeue operation is performed with respect to the queue descriptor, or an enqueue operation is performed with respect to the queue descriptor while the queue is empty.

23. The apparatus of claim 21 wherein the processor is configured to set the second valid bit if an enqueue operation is performed with respect to the queue descriptor.

24. The apparatus of claim 21 wherein the processor is configured to set a pointer's valid bit when the pointer is fetched from the memory to the cache.

25. The apparatus of claim 21 wherein the processor is configured to return to the memory the pointers whose valid bits has been set.

26. An article comprising a computer-readable medium that stores computer-executable instructions for causing a computer system to:

store in memory a plurality of queue descriptors each including a head pointer pointing to a first element in a corresponding queue and a tail pointer pointing to a last element in the corresponding queue;

Applicant : Gilbert Wolrich et al.
Serial No. : 10/039,289
Filed : January 4, 2002
Page : 21 of 24

Attorney's Docket No.: 10559-612001 / P12851

in response to a command to perform an enqueue or dequeue operation with respect to a first queue corresponding to a first queue descriptor, fetch from memory to a cache one of either a head pointer of the first queue descriptor pointing to a first element in the first queue or a tail pointer of the first queue descriptor pointing to a last element in the first queue; and

return to the memory from the cache the portions of the first queue descriptor modified by the operation.

27. The article of claim 26 including instructions to cause the computer system to:

fetch the head pointer and not the tail pointer of the first queue descriptor in response to a command to perform a dequeue operation with respect to the first queue; or

fetch the tail pointer and not the head pointer of the first queue descriptor in response to a command to perform an enqueue operation with respect to the first queue.

28. The article of claim 26 including instructions to cause the computer system to return to memory:

the head pointer and not the tail pointer of the first queue descriptor if only dequeue operations are performed on the first queue;

the tail pointer and not the head pointer of the first queue descriptor if only enqueue operations are performed on the first queue while the first queue is unempty; or

both the head pointer and tail pointer of the first queue descriptor if both an enqueue and a dequeue are performed on the first queue, or an enqueue operation was performed on the first queue while the first queue is empty.

29. The article of claim 26 including instructions to cause the computer system to set a valid bit corresponding to a pointer in the cache when the pointer is modified by an enqueue or a dequeue operation.

30. The article of claim 29 including instructions to cause the computer system to return to the memory pointers whose corresponding valid bits are set.

31.   A method comprising:

storing in memory a queue descriptor for a queue, the queue descriptor including a count identifying a number of elements in the queue, a head pointer pointing to a first element in the queue and a tail pointer pointing to a last element in the queue;

in response to a command to perform an enqueue or dequeue operation with respect to the queue, fetching from the memory to a cache the count and one of either the head pointer or tail pointer; and

returning to the memory from the cache portions of the queue descriptor modified by the operation.

32.   The method of claim 31 including fetching the count and the head pointer and not the tail pointer in response to a command to perform a dequeue operation; or fetching the count and the tail pointer and not the head pointer in response to a command to perform an enqueue operation.

33.   A method comprising:

determining whether a head pointer or a tail pointer of a queue descriptor that was fetched from memory to a cache in response to an enqueue or dequeue operation on a queue had been modified by the enqueue or dequeue operation; and

returning a count identifying a number of elements in the queue and one of either the head pointer or tail pointer to the memory from the cache only if that pointer had been modified.

34.   An apparatus comprising:

memory for storing queue descriptors which include a count identifying a number of elements in a queue, a head pointer pointing to a first element in the queue and a tail pointer pointing to a last element in the queue;

a cache for storing queue descriptors corresponding to up to a number of the memory's queue descriptors; and

a processor configured to:

fetch from the memory to the cache the count and one of either the head pointer or

the tail pointer of a particular queue descriptor in response to a command to perform an

enqueue or a dequeue operation with respect to the particular queue descriptor; and

return to the memory from the cache portions of the queue descriptor modified by

the operation.

35.   The apparatus of claim 34 wherein the processor is configured to fetch the count

and the head pointer and not the tail pointer in response to a command to perform a dequeue

operation; or fetch the count and the tail pointer and not the head pointer in response to a

command to perform an enqueue operation.

36.   An article comprising a computer-readable medium that stores computer-

executable instructions for causing a computer system to:

store in memory a queue descriptor including a count identifying a number of elements in

a queue, a head pointer pointing to a first element in the queue and a tail pointer pointing to a last

element in the queue;

in response to a command to perform an enqueue or dequeue operation with respect to a

queue, fetch from memory to a cache the count and one of either a head pointer pointing to a first

element in the queue or a tail pointer pointing to a last element in the queue; and

return to the memory from the cache the portions of the queue descriptor modified by the

operation.

37.   The article of claim 36 including instructions to cause the computer system to:

fetch the count and the head pointer and not the tail pointer in response to a command to

perform a dequeue operation; or

fetch the count and the tail pointer and not the head pointer in response to a command to

perform an enqueue operation.

Evidence Appendix

None

Related Proceedings Appendix

None